License Plate Recognition algorithm

This document describes a Computer Vision of the license plate recognition (LPR). This algorithm divided 3 process. “Plate Detect” process, “Character Segmentation” process and “Character Recognition” process. I'll explain this algorithm written by OpenCV in step-by-step.

Plate Detect

This algorithm has detected the license plate shape from the picture. I apply Gaussian filter, edge detect filter using Canny algorithm, and find contour algorithm. And I verification this algorithm to crop correct plate shape from images.

Before apply find contour algorithm, I need to convert color map to gray scale and apply Gaussian filter to detect edge. Because the image including many noise (by the camera or small dust.) and unnecessary parts. After loaded image, I will apply Gaussian filter using 5 x 5 operator. Gaussian filter is a filter that strongly weights to the example pixel. Following equation and filter operator to be used.

$$G(x,y) = \frac{1}{2\pi\sigma^2} exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)$$

$\frac{1}{256}$	$\frac{4}{256}$	$\frac{6}{256}$	$\frac{4}{256}$	$\frac{1}{256}$
$\frac{4}{256}$	$\frac{16}{256}$	$\frac{24}{256}$	$\frac{16}{256}$	$\frac{4}{256}$
$\frac{6}{256}$	$\frac{24}{256}$	$\frac{36}{256}$	$\frac{24}{256}$	$\frac{6}{256}$
$\frac{4}{256}$	$\frac{16}{256}$	$\frac{24}{256}$	$\frac{16}{256}$	$\frac{4}{256}$
$\frac{1}{256}$	$\frac{4}{256}$	$\frac{6}{256}$	$\frac{4}{256}$	$\frac{1}{256}$

($\sigma = 1.0$)

```
//Convert to grayscale
cv::cvtColor(src_img, src_gray, CV_RGB2GRAY);

//Gaussian filter (Noise reduction)
cv::GaussianBlur(src_gray, src_gray, cv::Size(5,5), 0);
```

After Gaussian filter, I will apply Canny edge detect filter. Canny method includes four steps.

1. Gaussian filter
2. Calculate intensity gradient
3. Non-maximum suppression (Thinning process)
4. Hysteresis threshold

Gaussian filter using parameter $\sigma = 1.4$. To calculate intensity gradient from image, I will use Sobel filter of X-axis and Y-axis. Used following operator and calculate strength of intensity gradient.

-1	0	1
-2	0	2
-1	0	1

X-axis Sobel operator: fx

-1	-2	-1
0	0	0
1	2	1

Y-axis Sobel operator: fy

$$Strength(x,y) = \sqrt{fx^2 + fy^2}$$

After calculating strength, the image has binarized by hysteresis threshold parameter. The threshold is determined by the difference of these parameters. To exclude unnecessary parts and to leave strong edge by license plate only, this parameter set strongly.

```
//Edge detection (Use canny method)
cv::Canny(src_gray, dst_img1, 150, 255);
```

The findContour function will give the contours of object from binary image. To find contours, define the Satoshi Suzuki and Keiichi Abe [Suzuki85]

algorithm. This function can be divided connected-component. I need to get external contours.

```
    //Find the contours in image
    cv::findContours(dst_img2, strage, CV_RETR_EXTERNAL,
CV_CHAIN_APPROX_TC89_KCOS);
```

The following code looks for a rectangle close to the license plate from the found contours. To find the rectangle object, using boundingRect function. This function will find the rectangle that fits on target object. After the find rectangle, the variable "rect" include information on the height and width of the rectangle. To calculate aspect ratio of the rectangle using this variable. Aspect ratio is different by Euro plate and old ones. I got this threshold parameter experimentally.

```
    //Search area of the plate with the correct aspect ratio
    for (int i = 0; i < strage.size(); i++) {
        size_t count = strage[i].size();
        if (count > 0) {
            cv::Mat pointr;
            cv::Rect rect;
            cv::Mat(strage[i]).convertTo(pointr, CV_32F);
            rect = cv::boundingRect(pointr);

            //Calculate aspect ratio
            prate = rect.width/rect.height;
            if (prate >=3 && prate <4.5 && rect.width < 250 && rect.width
> 150){

                //Write rectangle to original image
                cv::rectangle(src_img, cv::Point(rect.x,rect.y), cvPoint
(rect.x + rect.width, rect.y + rect.height), CV_RGB (0, 255, 0), 2);

                //Crop plate image
                dst_img3 =
src_gray(cv::Rect(rect.x,rect.y,rect.width,rect.height));
                break;
            }
        }
    }
```

Character segmentation

After the plate detect process, To find the character area, using same function when the function use in plate detection. But there are some

different in this function. Previous function use ".strage()". But this function use ".begin()" and ".end()" because when the findContours function use, contain the information of size of contours. But this information lined up by size, I can't get order of character image. Next step that search the aspect rate is same as prate detect process. To use in character recognition process, reject small objects (like as noise and "-" character in Finnish number) and too large objects (like as car grill and bumper).

```
   cv::findContours(dst_img4, strage, CV_RETR_EXTERNAL,
CV_CHAIN_APPROX_NONE);

    int cnt = 0;
    int offs2 = 1;
    char name[10];

    itc = strage.begin();

    //Compare with aspect rate in all contours
    while (itc!= strage.end()) {
              //Extract only current contour.
              std::vector<cv::Point> pointr = *itc;
              cv::Mat rectMat = cv::Mat(pointr);

              //Found the rectange surrounding the contour.
              cv::Rect rect = cv::boundingRect(rectMat);

              //Calculate the aspect rate. Character aspect rate < 1 and
Character size > 30
              prate = (double)rect.width/rect.height;
              if (prate >0 && prate <1 && itc -> size() > 30){

                  //Crop and resize the character to match the template.
                  cnt++;
                  chr_tmp =
dst_img3(cv::Rect(rect.x-offs2,rect.y-offs2,rect.width+2*offs2,rect.he
ight+2*offs2));
                  chr_img[cnt] = cv::Mat::zeros(100, 60, CV_8U);
                  cv::resize(chr_tmp, chr_img[cnt], chr_img[cnt].size());

                  //Show the characters.
                  std::sprintf(name, "chr_img%d",cnt);
                  std::string wname(name);
                  cv::namedWindow(wname, CV_WINDOW_AUTOSIZE);
                  cv::imshow(wname, chr_img[cnt]);

       }
       itc++;
    }
    cv::namedWindow("dst_img3", CV_WINDOW_AUTOSIZE);
```

```
    cv::imshow("dst_img3", dst_img3);
```

Character recognition

After the character segmentation, To character recognition, use the matchTemplate function. This function include default template matching algorithm and equation of correlation methods. In this case, use NCC (Normalized cross correlation) method. NCC method is strong for brightness. It used following equation. This equation show the result within the parameter of -1 to +1 because it use the cosine.

$$R_{NCC} = \frac{\sum_{j=0}^{N-1}\sum_{i=0}^{M-1} I(i,j)T(i,j)}{\sqrt{\sum_{j=0}^{N-1}\sum_{i=0}^{M-1} I(i,j)^2 \times \sum_{j=0}^{N-1}\sum_{i=0}^{M-1} J(i,j)^2}}$$

```
cv::Mat result;
    double tmpdata;
    double tmpmax = 0;
    int tmploc[cnt];
    char output[10];
    std::string out_plate;

    //Template matching
    for (int j = 1; j < cnt+1; j++) {
        for (int i = 1; i < 35; i++) {
            cv::matchTemplate(chr_img[j], tmp_chr[i], result,
CV_TM_CCOEFF_NORMED);
            tmpdata = result.at<float>(0,0);
            std::cout << j << ", " << i << ", " << tmpdata << std::endl;

            if (tmpdata > tmpmax) {
                tmpmax = tmpdata;
                tmploc[j] = i;
            }
        }
        if (tmpmax < 0.5) {
            tmploc[j] = 0;
        }
        tmpmax = 0;
    }
```

The following code means to convert between template number and character. This code compare with character code. In this algorithm, it is not possible to distinguish between “O” and “0”(Zero), “I” and “1”(One) Euro

plate. It excluded such as “-”(hyphen) specific to the license plate of Finland.

```
for (int j = cnt; j > 0; j--) {
       output[j] = tmploc[j];
       if (output[j] > 10) {
           //19~23
           if (output[j] > 18 && output[j] < 24) {
               output[j] = 'A' + tmploc[j]-10;
           //24~34
           }else if (output[j] > 23){
               output[j] = 'A' + tmploc[j]-9;
           }else{
           //11~18
           output[j] = 'A' + tmploc[j]-11;
           }
           //10
       }else if (tmploc[j] == 10){
           output[j] = 48;
           //1~9
       }else{
           output[j] = tmploc[j] + 48;
       }
       //0
       if (tmploc[j] > 0) {
           out_plate += output[j];
       }
   }
```